Customer Churn

In [ ]:
```
#importing required Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
print("Libraries imported")
```

```
/Users/akashyadav/opt/anaconda3/lib/python3.9/site-packages/scipy/__init__.
py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for t
his version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
Libraries imported
```

Loading the Dataset

In [ ]:
```
#Dataset
telecom_base_data = pd.read_csv('Customer-Churn.csv')
print('Dats Read Success')
```

```
Dats Read Success
```

Checking top 5 Record

In [ ]:
```
telecom_base_data.head()
```

Out[ ]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multiple
0	7590-VHVEG	Female	0	Yes	No	1	No	No p s
1	5575-GNVDE	Male	0	No	No	34	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	
3	7795-CFOCW	Male	0	No	No	45	No	No p s
4	9237-HQITU	Female	0	No	No	2	Yes	

5 rows × 21 columns

Checking the Number of Rows and column in Dataset

In [ ]:
```
telecom_base_data.shape
```

Out[ ]:
```
(7043, 21)
```

Checking Column Types

In [ ]:
```
telecom_base_data.columns.values
```

Out[ ]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

Checking Datatypes of Each Column

In [ ]:
```
telecom_base_data.dtypes
```

Out[ ]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

Checking the Descriptive Statics of Numerical Values of Dataset

In [ ]:
```
telecom_base_data.describe()
```

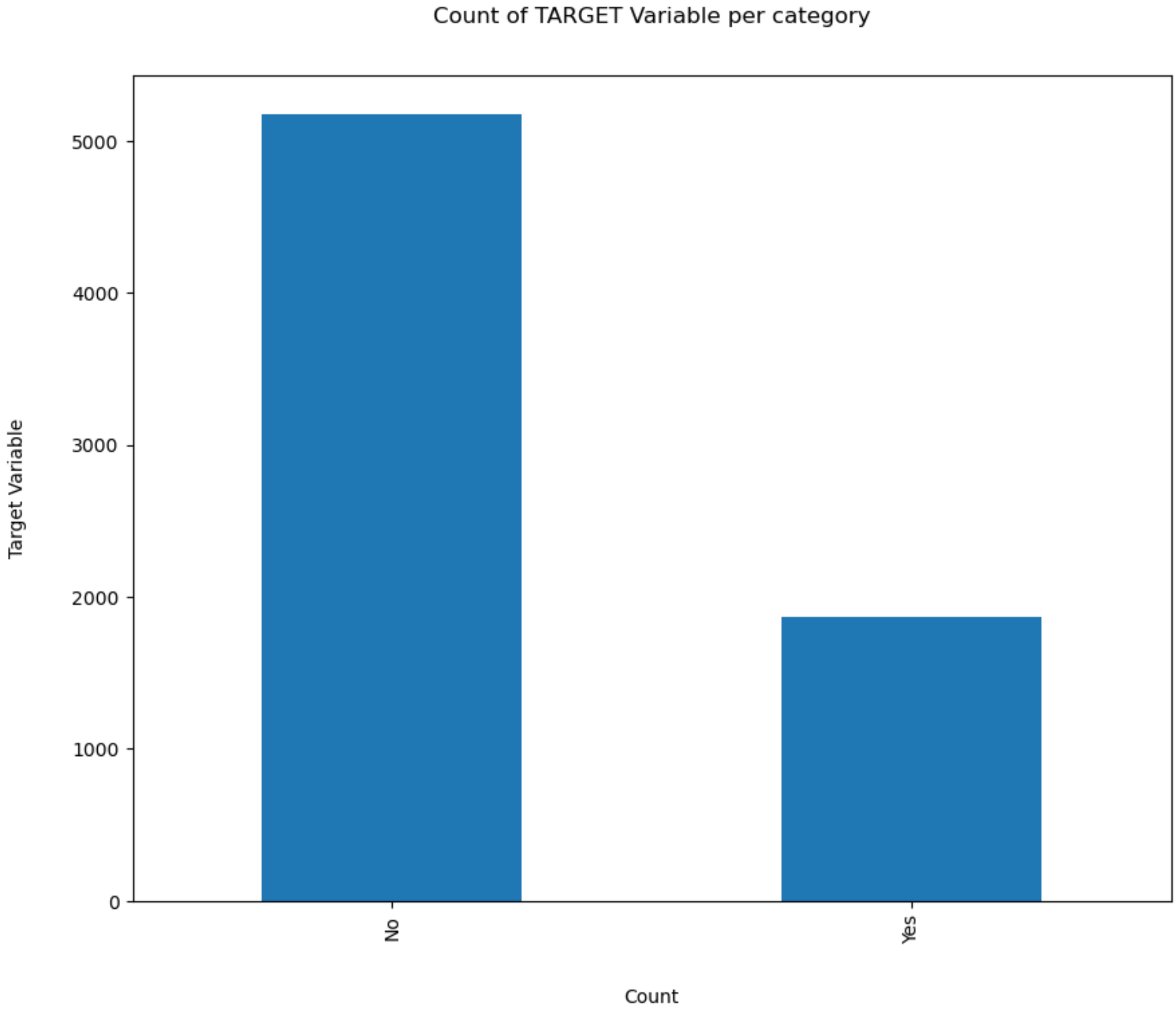
Out[ ]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

Insights of Our Dataset (Observed by Seeing above Output) Senior Citizens are categorical values that is why 25% 50% 75% are not Correct 25% Customers have tenure less than 9 Months 50% Customers have tenure less than 29 Months 75% Customers have tenure less than 55 Months Average Monthly charges are 64.76 USD

Now we visualize these information in to Graph/Plot etc. We will now classify whether the customer churn or not. So this is a Binary Clasification as Customer will Churn : Yes or No so we have to Analyse Yes:No Ratio

In [ ]:
```
#finding Churn to Non Churn Ratio
telecom_base_data['Churn'].value_counts().plot(kind='bar', figsize=(10, 8))
plt.xlabel("Count", labelpad=24)
plt.ylabel("Target Variable", labelpad=24)
plt.title("Count of TARGET Variable per category", y=1.05);
```



In [ ]:
```
#Getting Exact Values
100*telecom_base_data['Churn'].value_counts()/len(telecom_base_data['Churn']
```

Out[ ]:
```
No     73.463013
Yes    26.536987
Name: Churn, dtype: float64
```

In [ ]:
```
telecom_base_data['Churn'].value_counts()
```

Out[ ]:
```
No     5174
Yes    1869
Name: Churn, dtype: int64
```

Here we can notice that the Dataset is Imbalanced. Imbalanced means the ratio of Churn:NonChurn is not balanced the number of Non Churn is way more than Churn by analysing above data The Ratio is approx 74:26 which is not balanced

For Imbalanced data we can use Upsampling and Downsampling. UpSampling : We synthetically increase the record of lower class DownSampling : We synthetically decrease the record of upper class Upsampling is better than Downsampling because of more Records

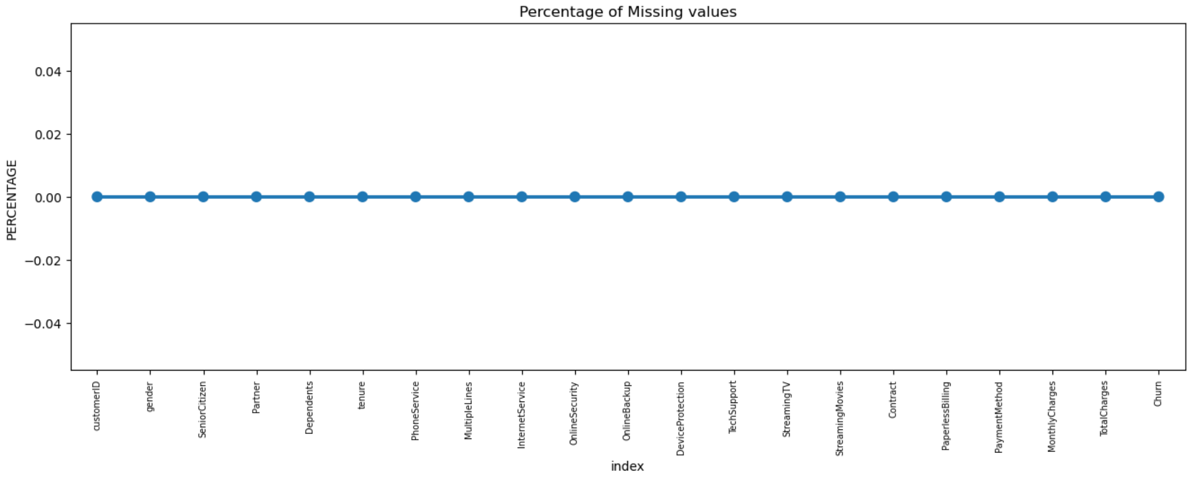
So we analyse the data with other features while taking the target values separately to get some insights.

In [ ]:
```
# Concise Summary of the dataframe, as we have too many columns, using the v
telecom_base_data.info(verbose = True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

Finding Percentage of Missing Values

In [ ]:
```
#Finding Missing Values Percentage
missing = pd.DataFrame((telecom_base_data.isnull().sum())*100/telecom_base_d
plt.figure(figsize=(16,5))
ax = sns.pointplot(x="index",y=0,data=missing)
plt.xticks(rotation =90,fontsize =7)
plt.title("Percentage of Missing values")
plt.ylabel("PERCENTAGE")
plt.show()
```



We can easily analyse that no attribute has null values in our Dataset

Now we Clean Our Data

In [ ]:
```
#creating a copy of Dataset
telecom_data = telecom_base_data.copy()
```

In [ ]:
```
telecom_data.dtypes
```

Out[ ]:
```
customerID          object
gender              object
SeniorCitizen        int64
Partner             object
Dependents          object
tenure               int64
PhoneService        object
MultipleLines       object
InternetService     object
OnlineSecurity      object
OnlineBackup        object
DeviceProtection    object
TechSupport         object
StreamingTV         object
StreamingMovies     object
Contract            object
PaperlessBilling    object
PaymentMethod       object
MonthlyCharges     float64
TotalCharges        object
Churn               object
dtype: object
```

Here TotalCharges column has object datatype so we will convert it to numeric datatype

In [ ]:
```
#from object to numeric
telecom_data.TotalCharges = pd.to_numeric(telecom_data.TotalCharges, errors=
#Calculating number of missing values in each column
telecom_data.isnull().sum()
```

Out[ ]:
```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

Here we can see that column TotalCharges have 11 missing values.

In [ ]:
```
#checking the records which have TotalCharges as null value
telecom_data.loc[telecom_data['TotalCharges'].isnull()==True]
```

Out[ ]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Mult
488	4472-LVYGI	Female	0	Yes	Yes	0	No	
753	3115-CZMZD	Male	0	No	Yes	0	Yes	
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	

11 rows × 21 columns

We have treat the Null value records There are 11 records out of 7043 records

In [ ]:
```
#percentage of Null records
print((11/7043)*100)
```

```
0.1561834445548772
```

0.156% is very less so we can drop column TotalCharges

In [ ]:
```
#dropping the record with nan value
telecom_data.dropna(how='any',inplace=True)
```

In [ ]:
```
#after drooping records
telecom_data.shape
```

Out[ ]:
```
(7032, 21)
```

Dividing customers into bins based on tenure like for tenure < 12 months: assign a tenure group if 1-12, for tenure between 1 to 2 Yrs, tenure group of 13-24

In [ ]:
```
# Getting the max tenure
print(telecom_data['tenure'].max())
```

```
72
```

Here the maximum Tenure is 72 Months so divide in to (1-12) months in one group and so on

In [ ]:
```
# Group the tenure in bins of 12 months
labels = ["{0} - {1}".format(i, i + 11) for i in range(1, 72, 12)]

telecom_data['tenure_group'] = pd.cut(telecom_data.tenure, range(1, 80, 12),
```

In [ ]:
```
telecom_data['tenure_group'].value_counts()
```

Out[ ]:
```
1 - 12     2175
61 - 72    1407
13 - 24    1024
25 - 36     832
49 - 60     832
37 - 48     762
Name: tenure_group, dtype: int64
```

Remove some columns which are not required for processing

In [ ]:
```
#dropping column customerID and tenure
telecom_data.drop(columns= ['customerID','tenure'], axis=1, inplace=True)
```

Checking data

In [ ]:
```
telecom_data.shape
```

Out[ ]:
```
(7032, 20)
```

In [ ]:
```
telecom_data.head()
```

Out[ ]:

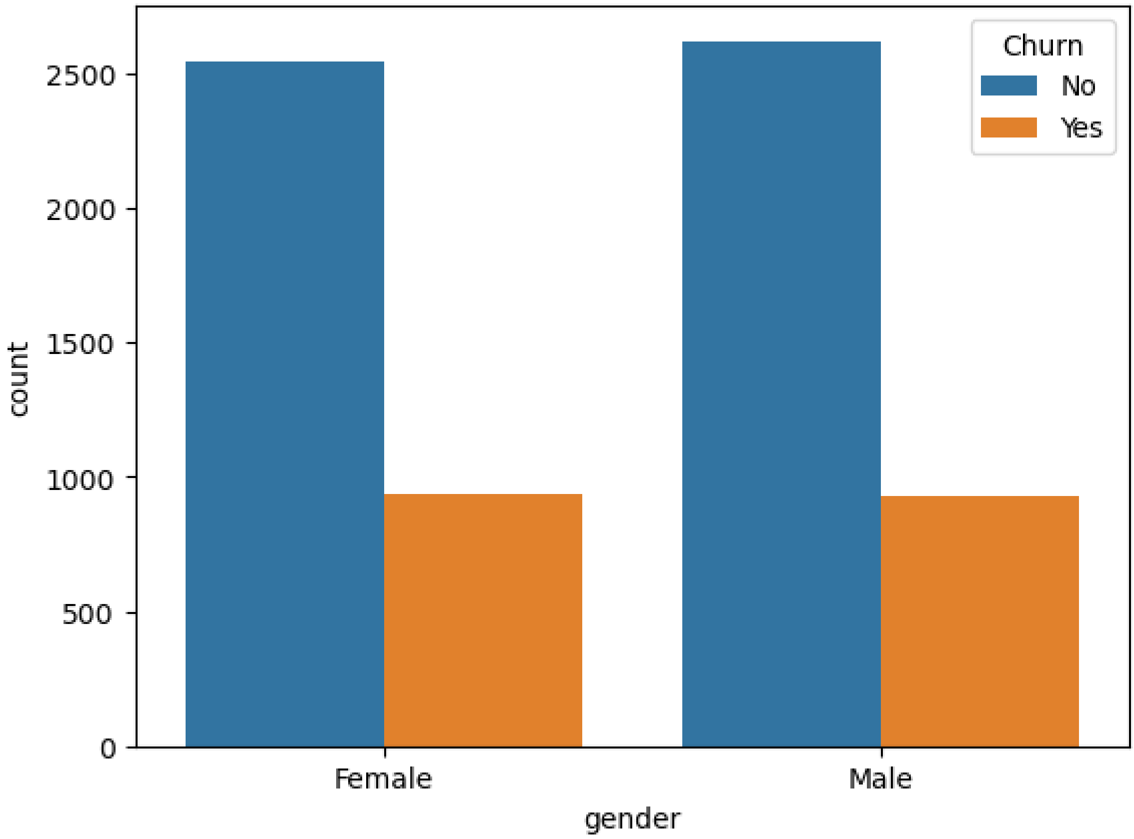
	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService
0	Female	0	Yes	No	No	No phone service	DSL
1	Male	0	No	No	Yes	No	DSL
2	Male	0	No	No	Yes	No	DSL
3	Male	0	No	No	No	No phone service	DSL
4	Female	0	No	No	Yes	No	Fiber optic

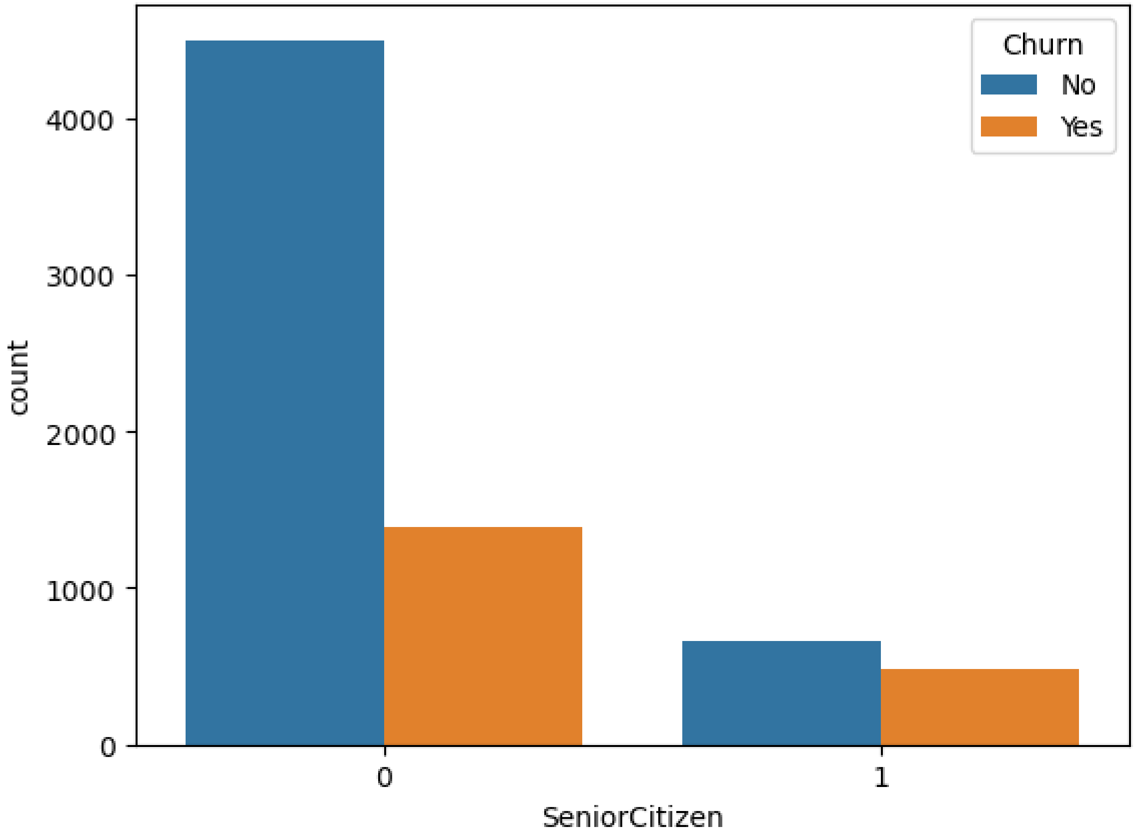
EDA (Exploratory Data Analysis)

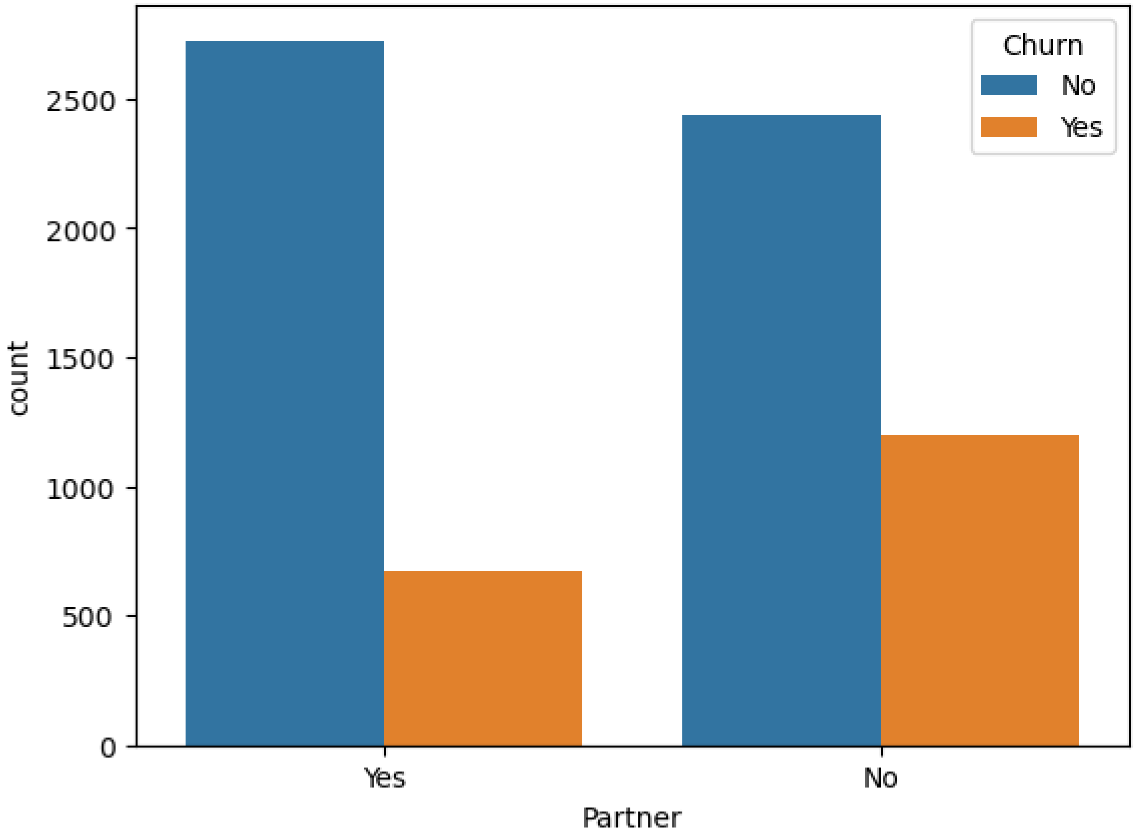
Creating Plot for Each column with the Churn

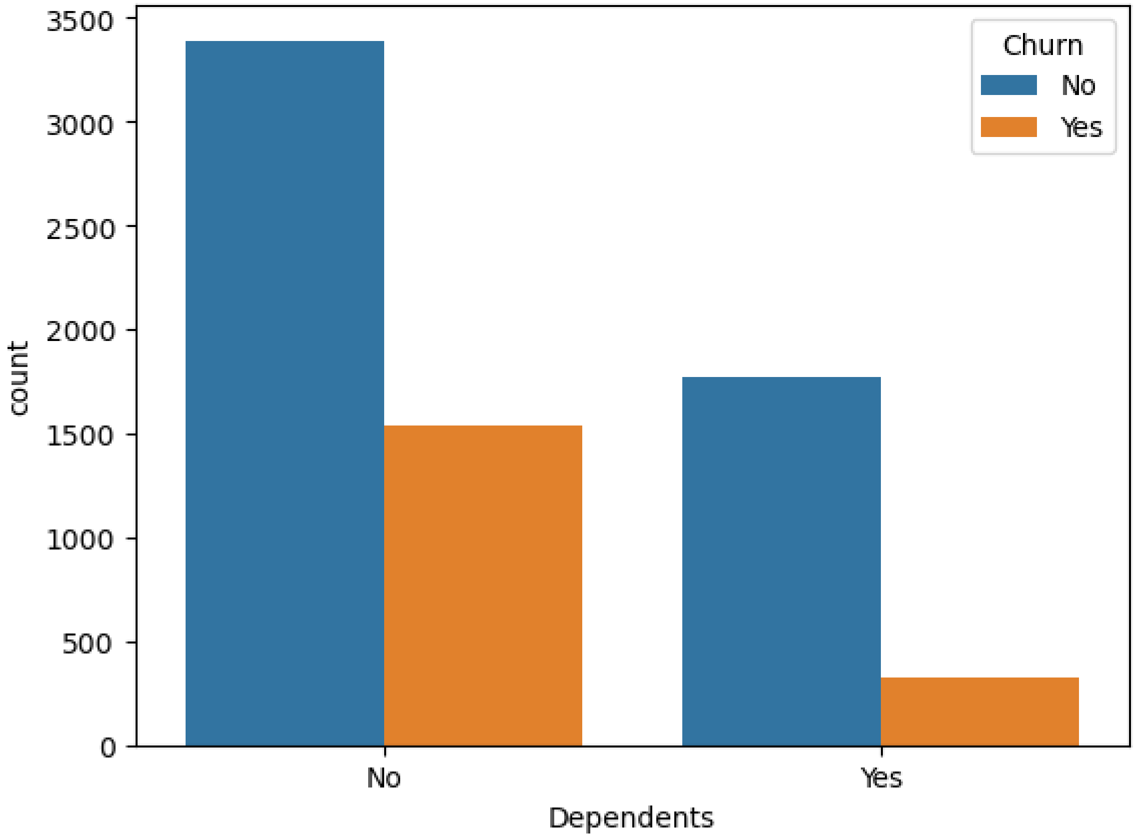
In [ ]:

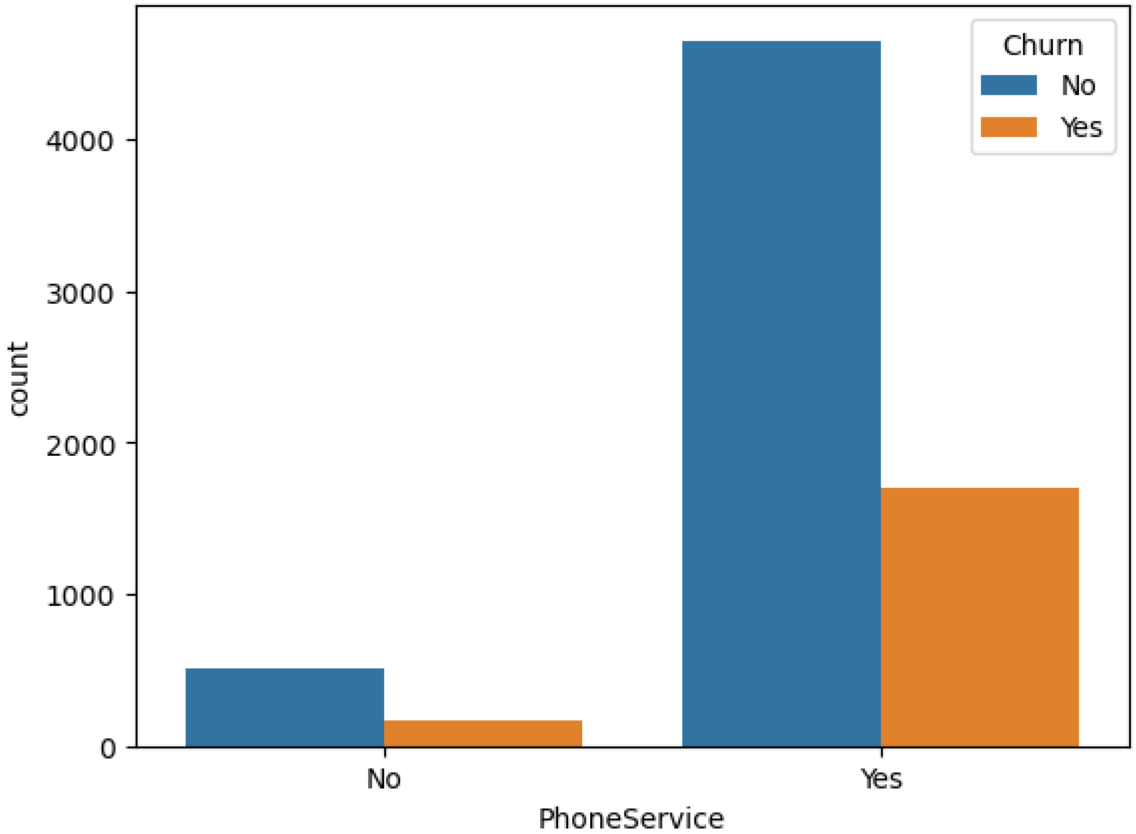
```
#Single Variable analysis
for i, predictor in enumerate(telecom_data.drop(columns=['Churn', 'TotalChar
    plt.figure(i)
    sns.countplot(data=telecom_data, x=predictor, hue='Churn')
```

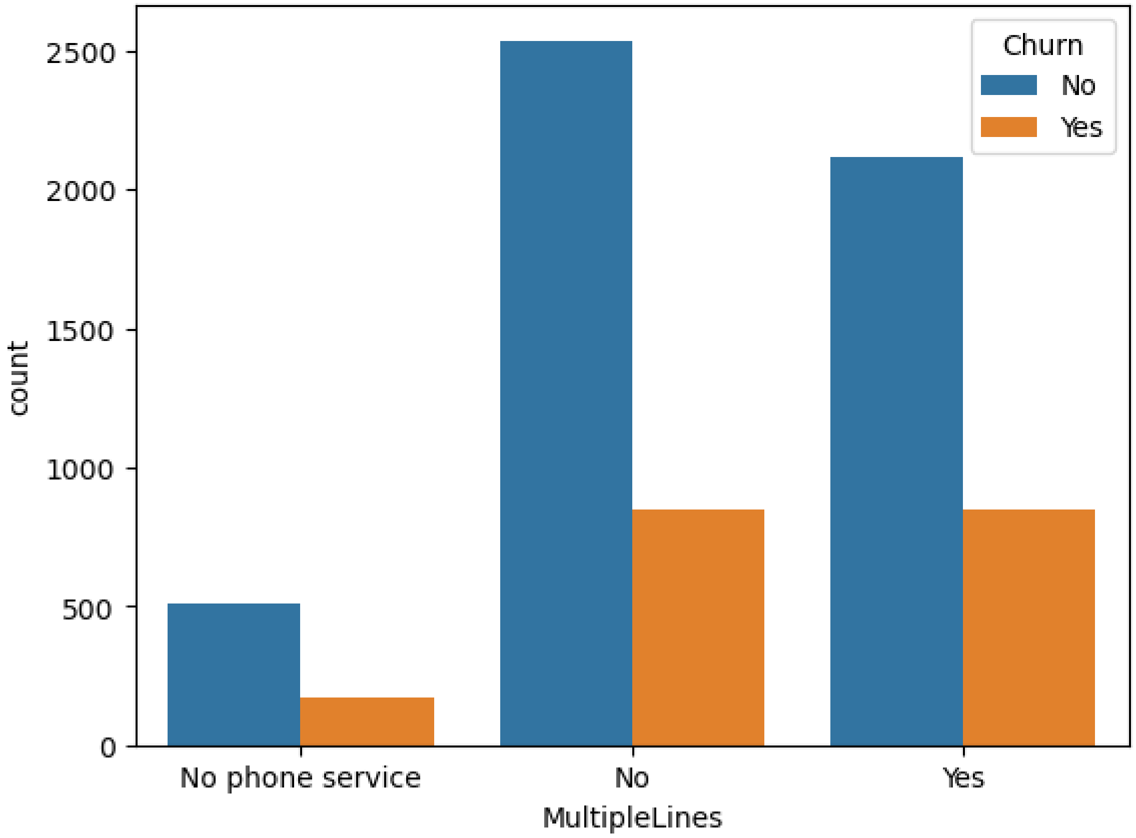


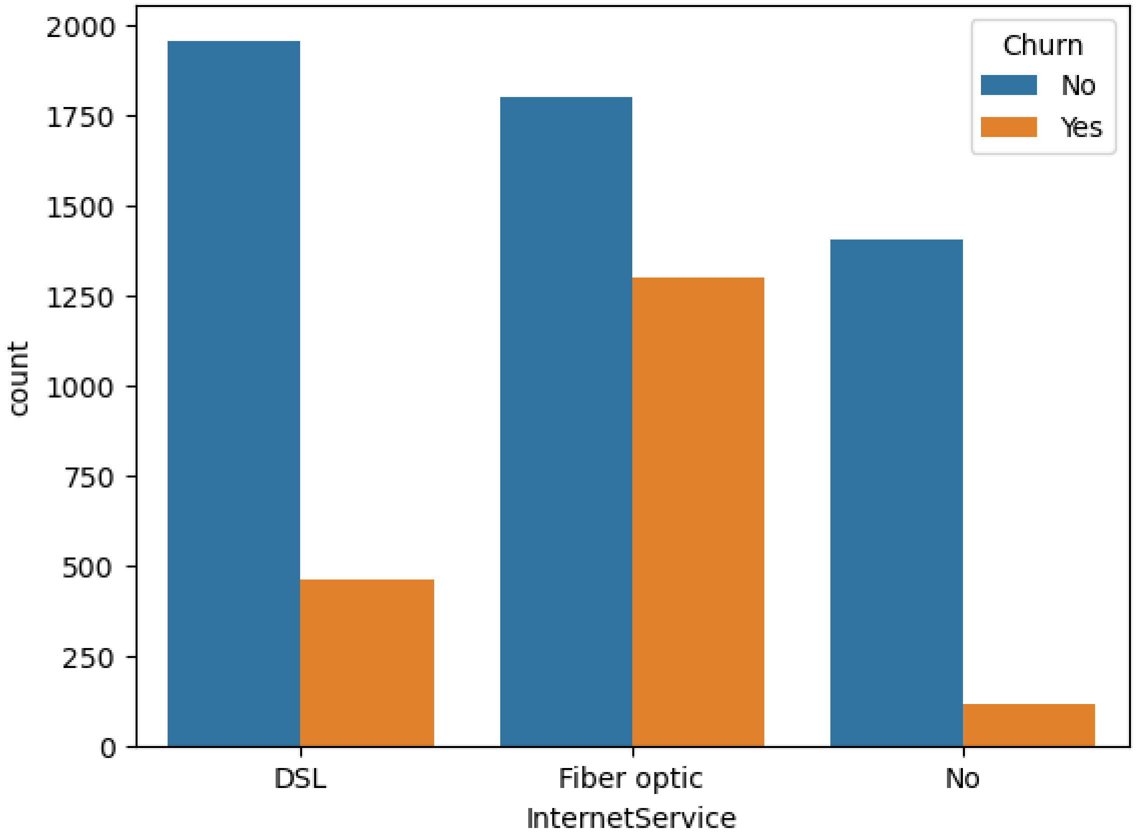

Churn
No
Yes
4000
3000
2000
1000
0
count
0
1
SeniorCitizen

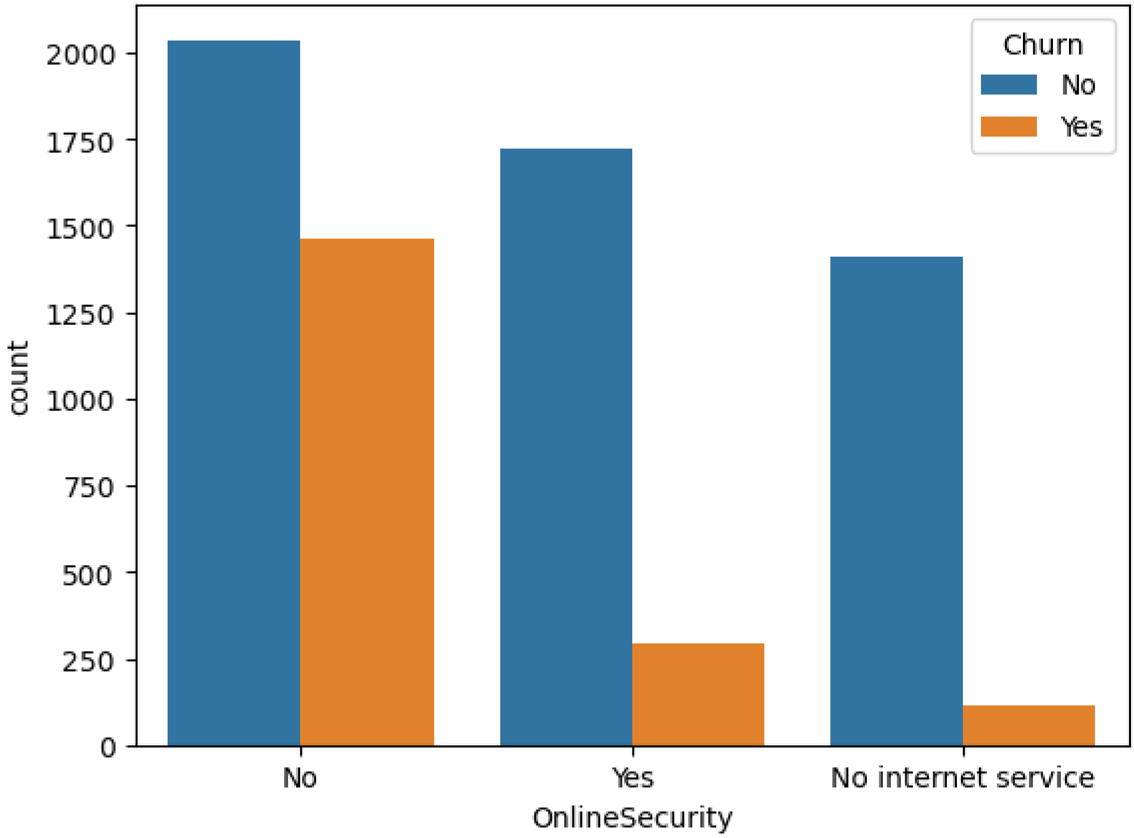

Churn
No
Yes
2500
2000
1500
1000
500
0
count
Yes
No
Partner

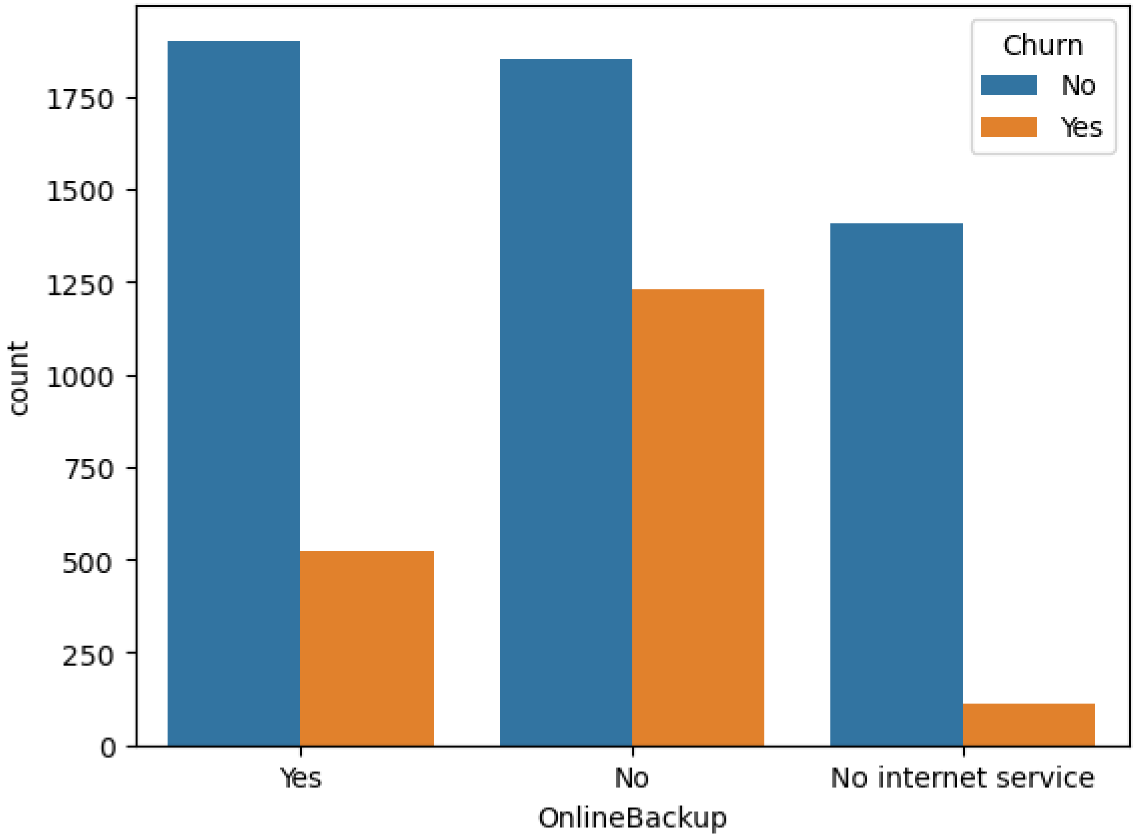

Churn
No
Yes
count
3500
3000
2500
2000
1500
1000
500
0
No
Yes
Dependents

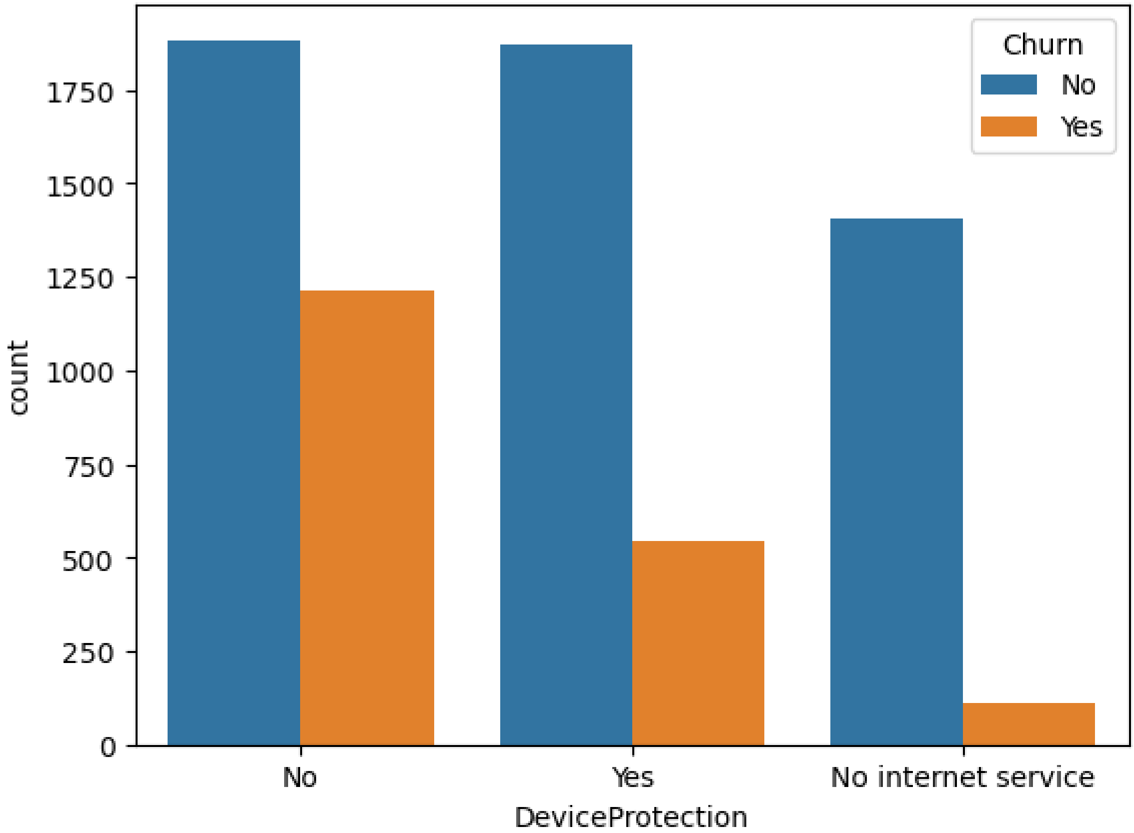

Churn
No
Yes
count
4000
3000
2000
1000
0
No
Yes
PhoneService


Churn
No
Yes
2500
2000
1500
1000
500
0
count
No phone service
No
Yes
MultipleLines

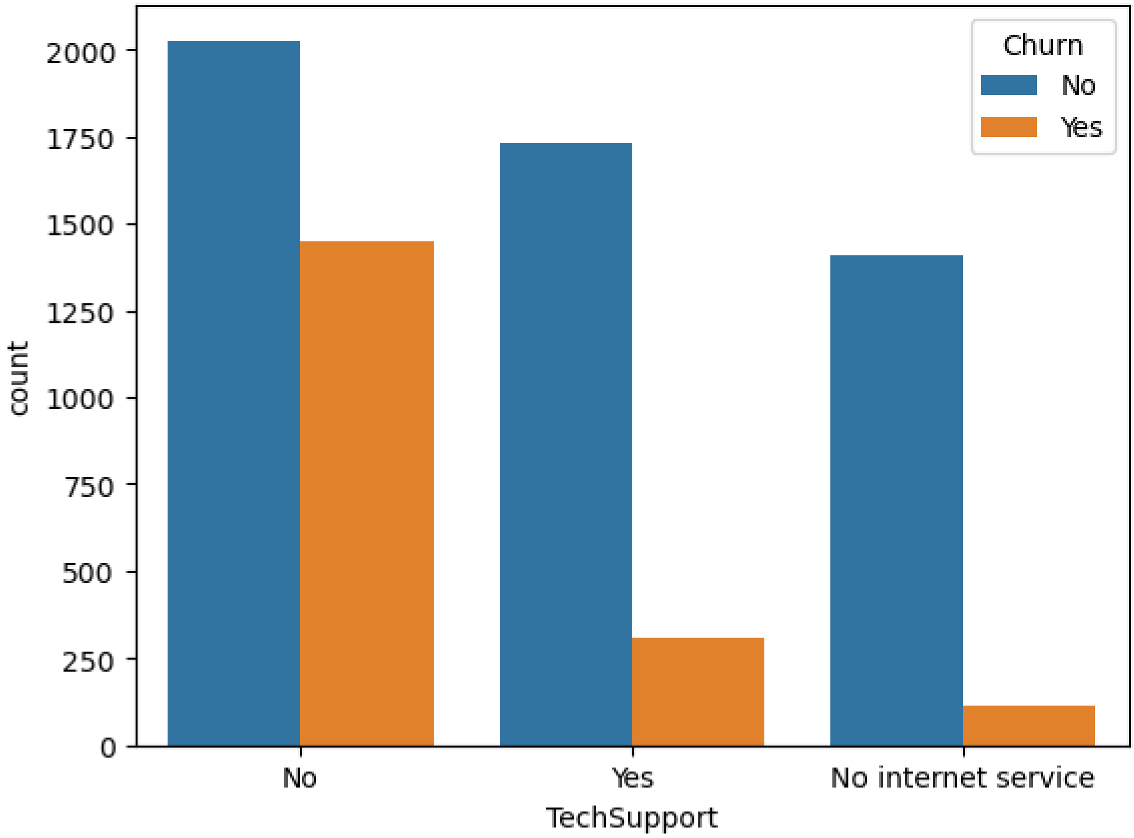

Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
DSL
Fiber optic
No
InternetService


Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
OnlineSecurity

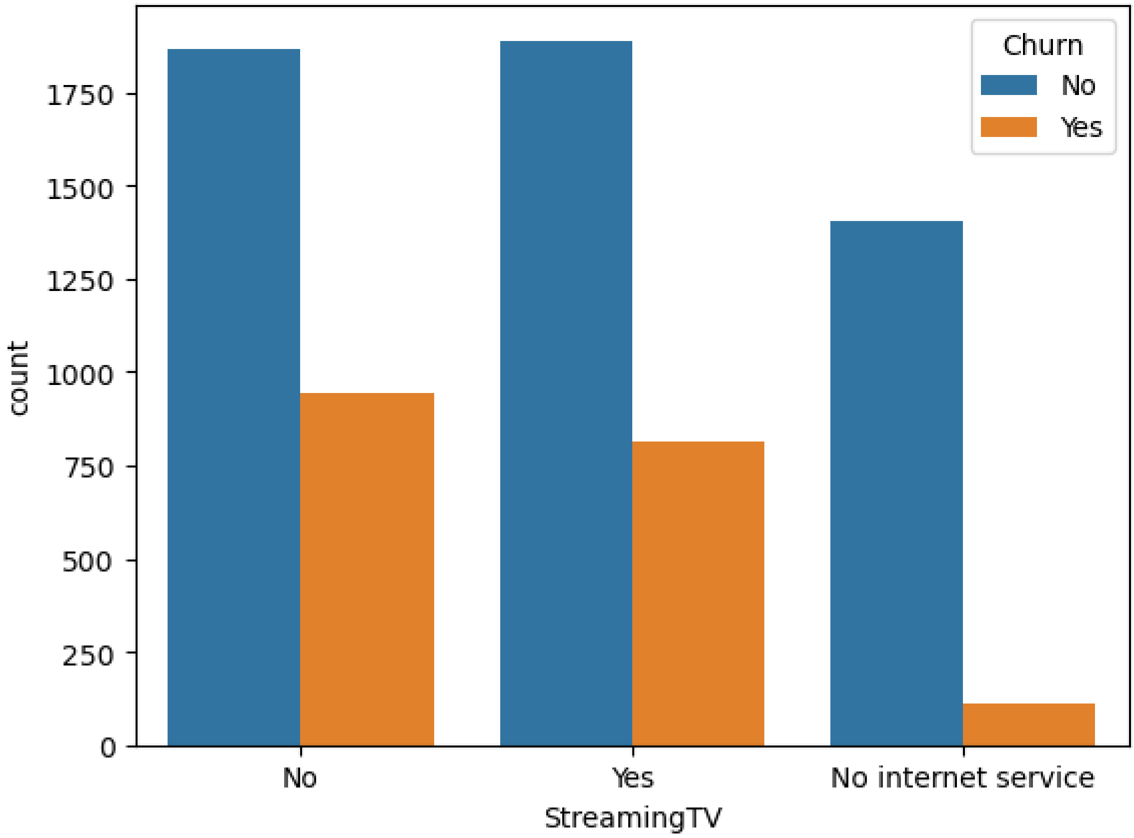

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
Yes
No
No internet service
OnlineBackup

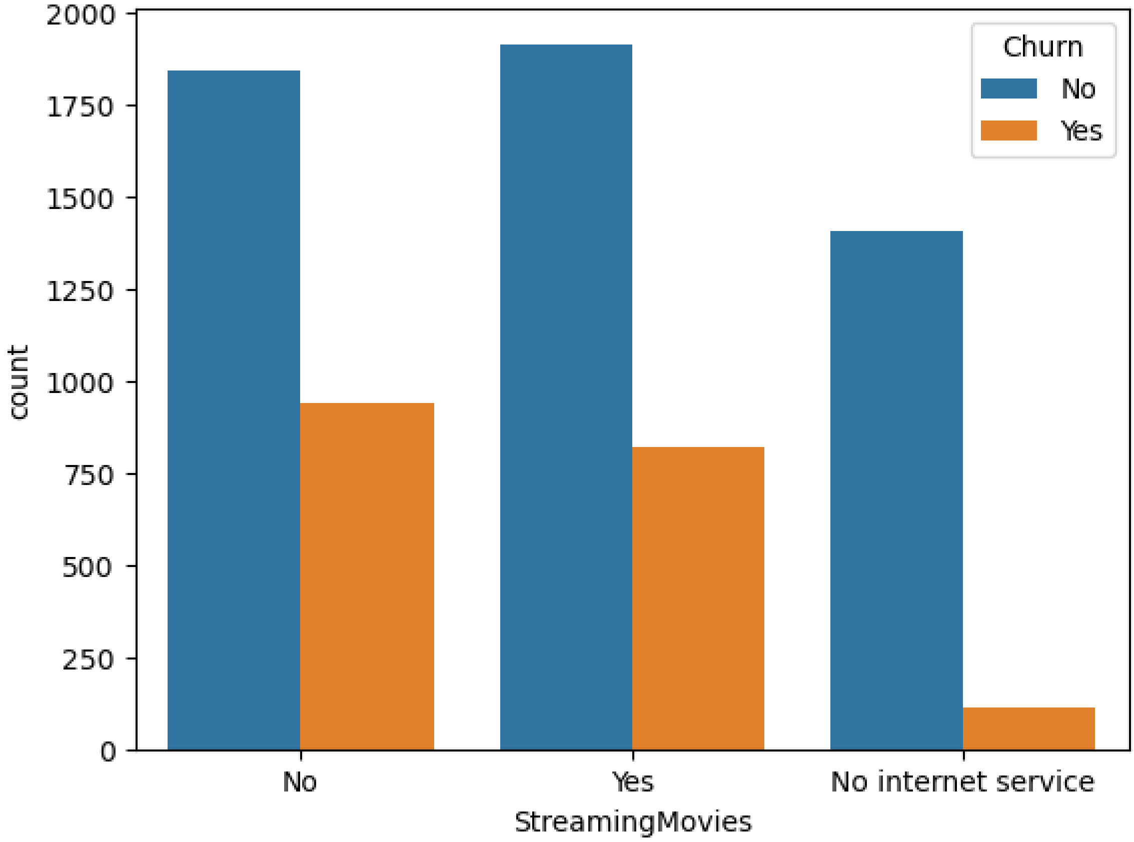


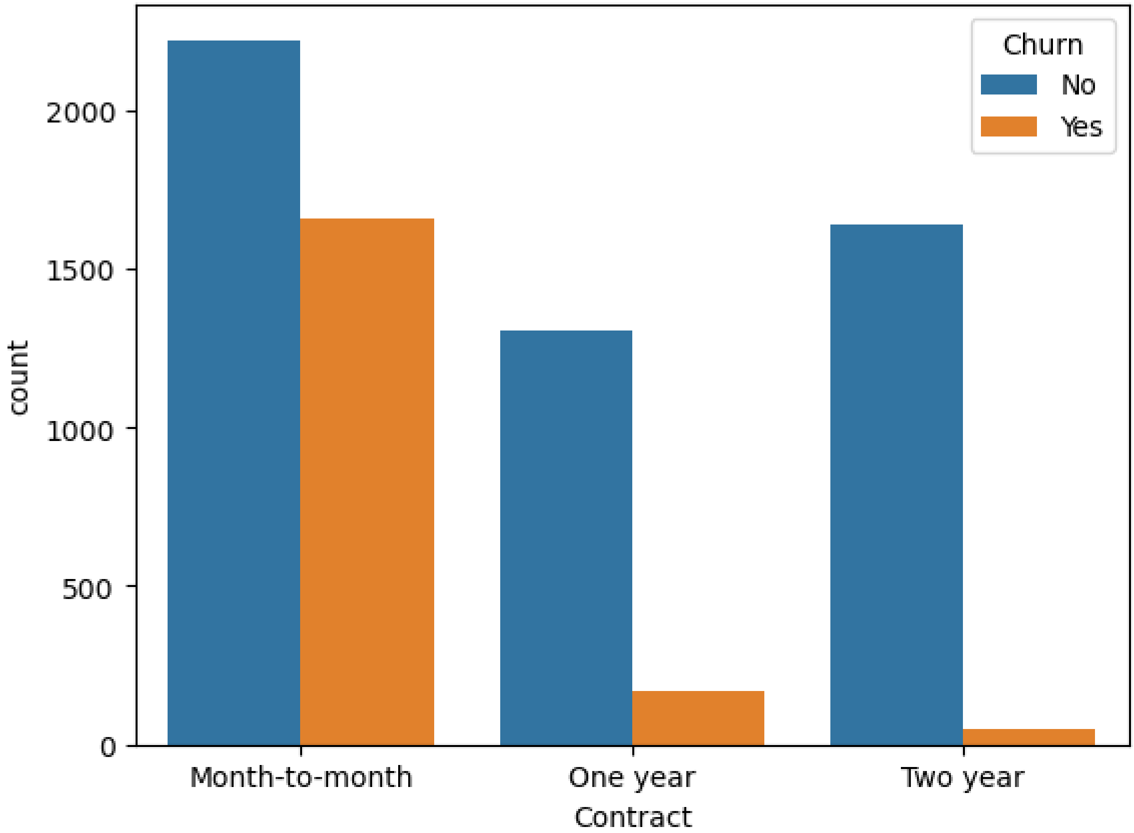
Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
No
Yes
No internet service
DeviceProtection

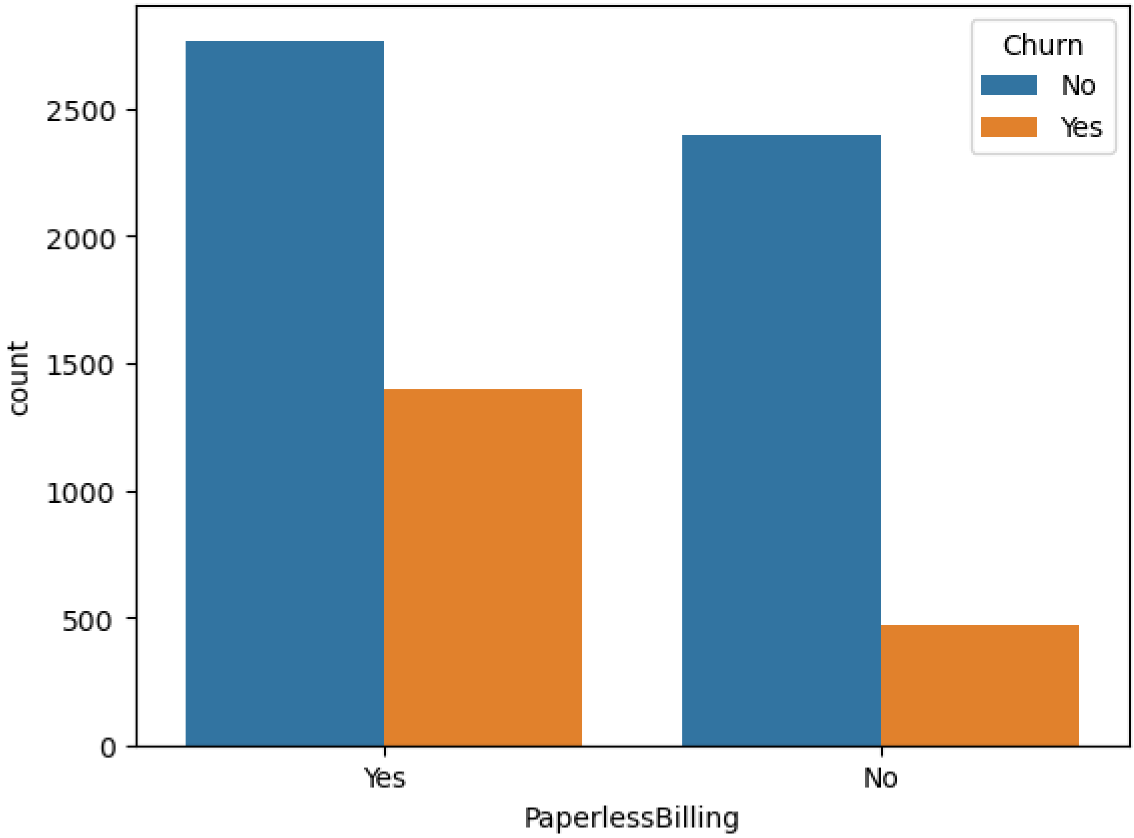


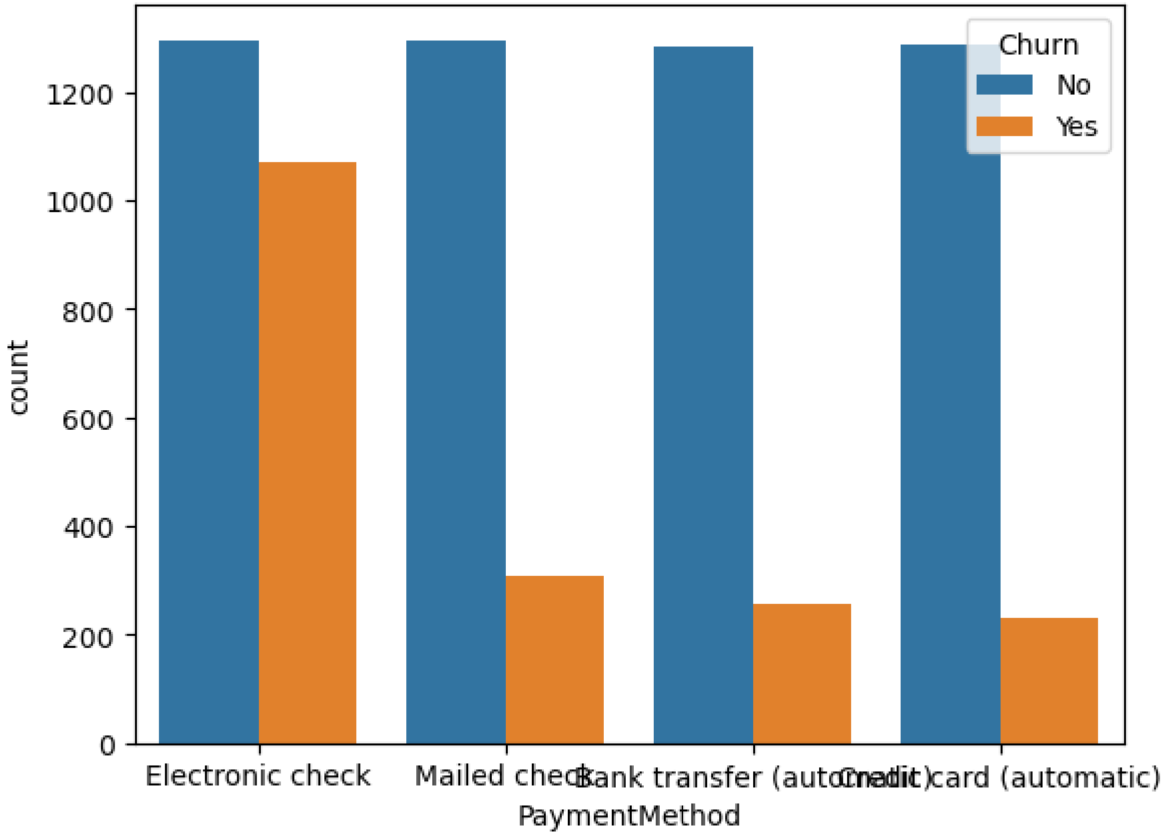
Churn
No
Yes
count
0
250
500
750
1000
1250
1500
1750
2000
No
Yes
No internet service
TechSupport

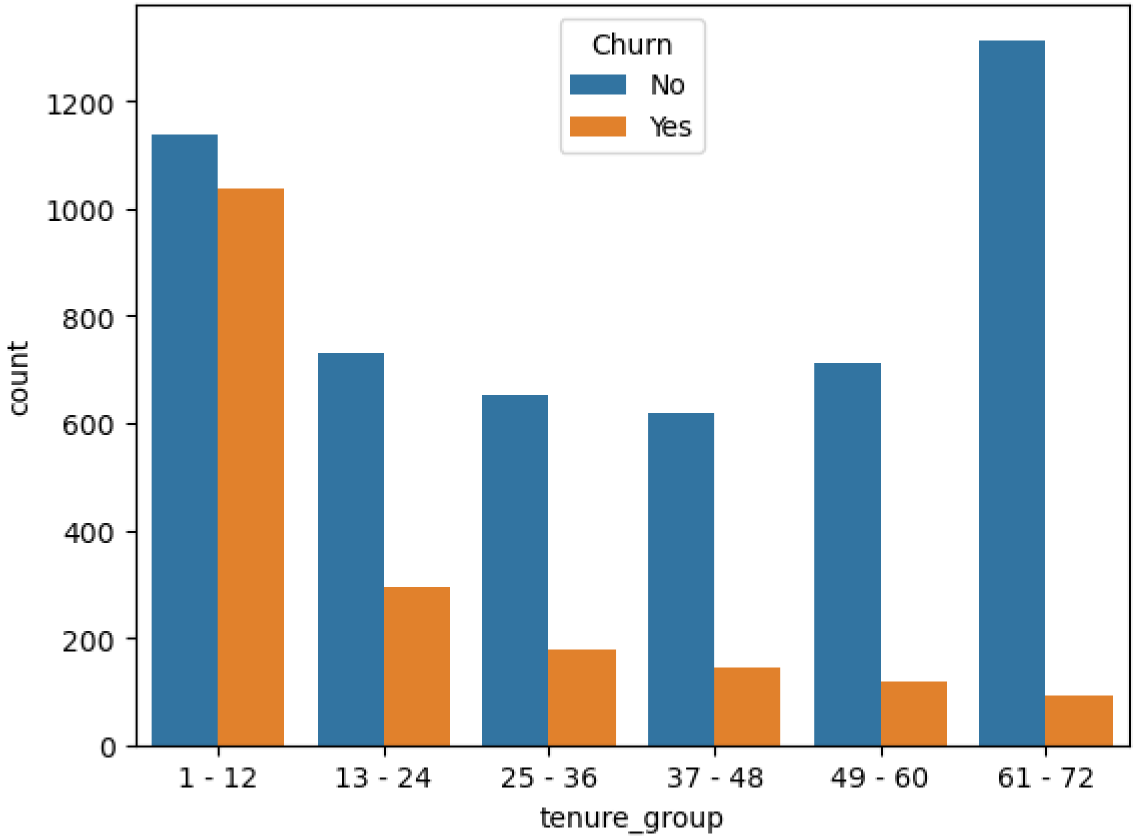

Churn
No
Yes
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
StreamingTV


Churn
No
Yes
2000
1750
1500
1250
1000
750
500
250
0
count
No
Yes
No internet service
StreamingMovies


Churn
No
Yes
2000
1500
1000
500
0
count
Month-to-month
One year
Two year
Contract


Churn
No
Yes
2500
2000
1500
1000
500
0
count
Yes
No
PaperlessBilling





Convert the target variable 'Churn' in a binary numeric variable i.e. Yes=1 ; No = 0

In [ ]:
```
#churn into 0 or 1
telecom_data['Churn'] = np.where(telecom_data.Churn == 'Yes',1,0)
```

In [ ]:
```
#checking our data
telecom_data.head()
```

Out[ ]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService
0	Female	0	Yes	No	No	No phone service	DSL
1	Male	0	No	No	Yes	No	DSL
2	Male	0	No	No	Yes	No	DSL
3	Male	0	No	No	No	No phone service	DSL
4	Female	0	No	No	Yes	No	Fiber optic

Converting all the categorical variables into dummy variables Using One Hot Encoding Example Gender Geography M Goa
F Mumbai M Bangalore F Goa F Delhi F Delhi

One Hot Encoding

Gender Geography Geo_Goa Geo_Mumbai Geo_Bangalore Geo_Delhi
M Goa 1 0 0 0 F Mumbai 0 1 0 0 M Bangalore 0 0 1 0 F Goa 1 0 0 0 F Delhi 0 0 0 1 F Delhi 0 0 0 1 This is One Hot Encoding

In [ ]:
```
#creating dummy variable
telecom_data_dummies = pd.get_dummies(telecom_data)
```

In [ ]:
```
#checking our data
telecom_data_dummies.head()
```

Out[ ]:

	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Parti
0	0	29.85	29.85	0	1	0	
1	0	56.95	1889.50	0	0	1	
2	0	53.85	108.15	1	0	1	
3	0	42.30	1840.75	0	0	1	
4	0	70.70	151.65	1	1	0	

5 rows × 51 columns

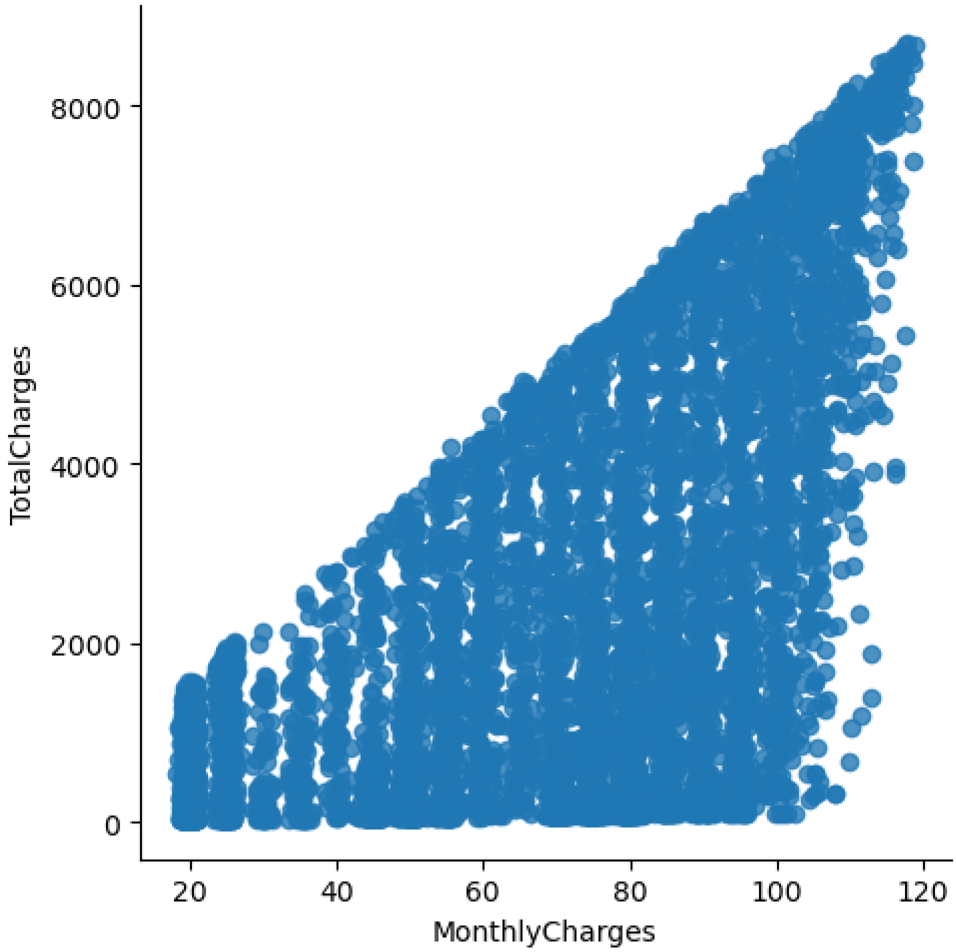
In [ ]:
```
telecom_data.shape
```

Out[ ]:
```
(7032, 20)
```

Creating relationship between MonthlyCharges and TotalCharges

In [ ]:
```
sns.lmplot(data=telecom_data_dummies, x='MonthlyCharges', y='TotalCharges',
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7fc7ba025e20>
```



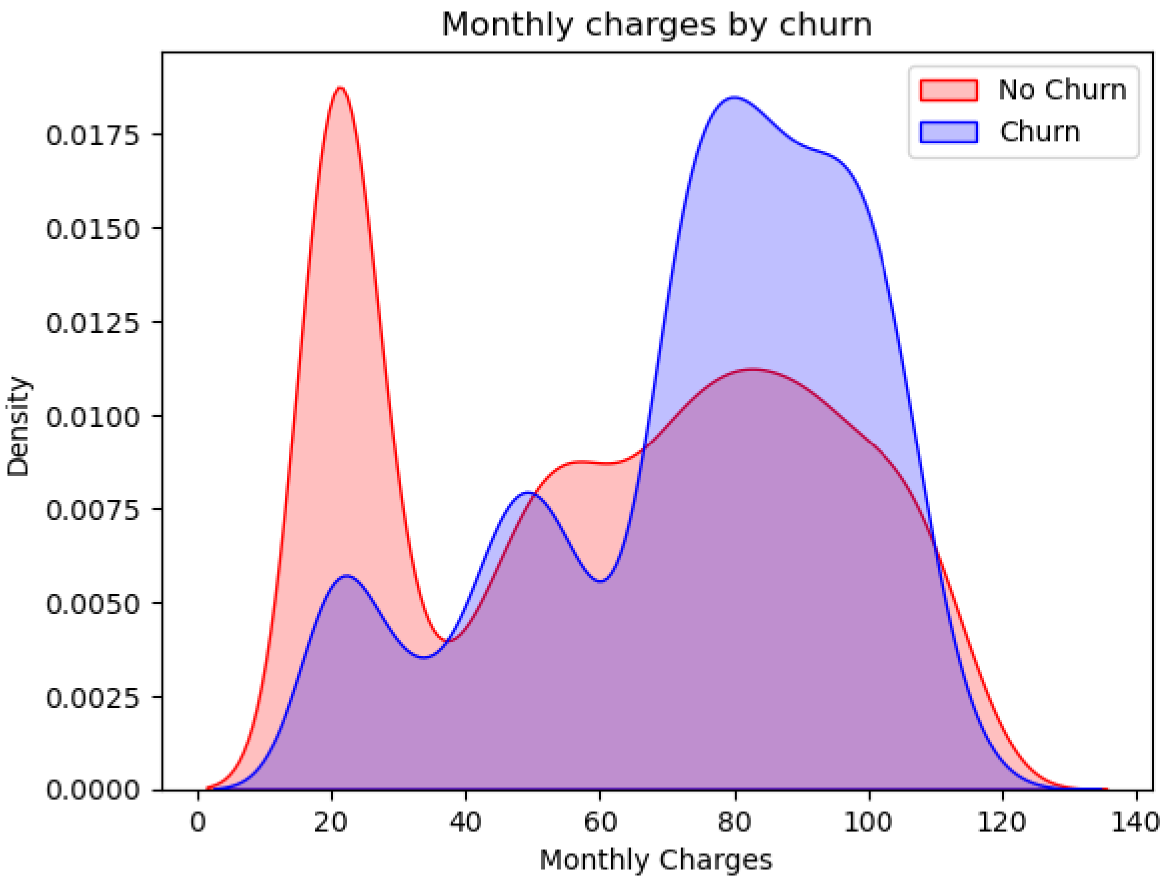
TotalCharges increases if MonthlyCharges were increased

Visualization of Data through Graphs

In [ ]:
```
#More Visualization
Mth = sns.kdeplot(telecom_data_dummies.MonthlyCharges[(telecom_data_dummies[
                color="Red", shade = True)
Mth = sns.kdeplot(telecom_data_dummies.MonthlyCharges[(telecom_data_dummies[
                ax =Mth, color="Blue", shade= True)
Mth.legend(["No Churn","Churn"],loc='upper right')
Mth.set_ylabel('Density')
Mth.set_xlabel('Monthly Charges')
Mth.set_title('Monthly charges by churn')
```

```
/var/folders/jq/x_2mslwx1p3_r8ps6dq_3j800000gn/T/ipykernel_23430/164977653
5.py:2: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Mth = sns.kdeplot(telecom_data_dummies.MonthlyCharges[(telecom_data_dummi
es["Churn"] == 0) ],
/var/folders/jq/x_2mslwx1p3_r8ps6dq_3j800000gn/T/ipykernel_23430/164977653
5.py:4: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Mth = sns.kdeplot(telecom_data_dummies.MonthlyCharges[(telecom_data_dummi
es["Churn"] == 1) ],
```

Out[ ]: Text(0.5, 1.0, 'Monthly charges by churn')

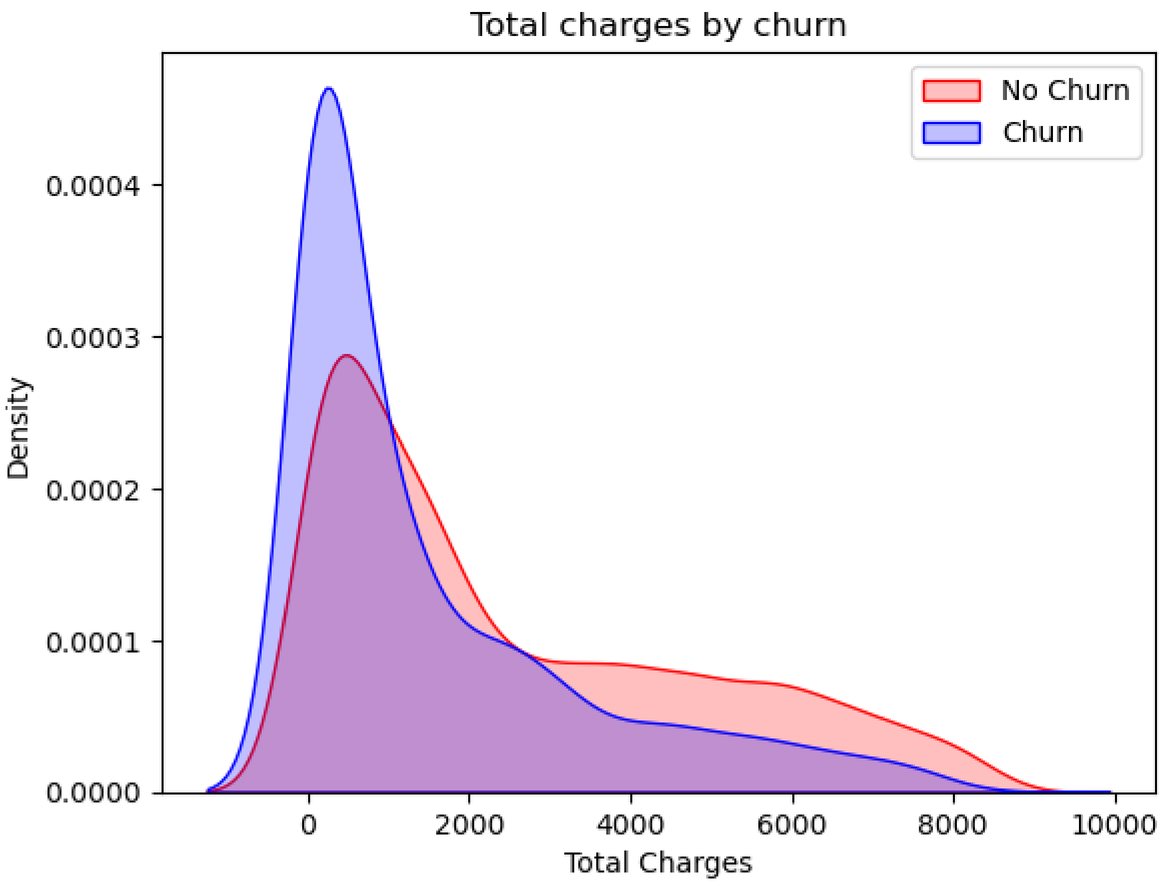


Insight: Churn is high when Monthly Charges ar high

In [ ]:
```
Tot = sns.kdeplot(telecom_data_dummies.TotalCharges[(telecom_data_dummies["C
                color="Red", shade = True)
Tot = sns.kdeplot(telecom_data_dummies.TotalCharges[(telecom_data_dummies["C
                ax =Tot, color="Blue", shade= True)
Tot.legend(["No Churn","Churn"],loc='upper right')
Tot.set_ylabel('Density')
Tot.set_xlabel('Total Charges')
Tot.set_title('Total charges by churn')
```

```
/var/folders/jq/x_2mslwx1p3_r8ps6dq_3j800000gn/T/ipykernel_23430/121370859
7.py:1: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Tot = sns.kdeplot(telecom_data_dummies.TotalCharges[(telecom_data_dummies
["Churn"] == 0) ],
/var/folders/jq/x_2mslwx1p3_r8ps6dq_3j800000gn/T/ipykernel_23430/121370859
7.py:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  Tot = sns.kdeplot(telecom_data_dummies.TotalCharges[(telecom_data_dummies
["Churn"] == 1) ],
```

Out[ ]: Text(0.5, 1.0, 'Total charges by churn')



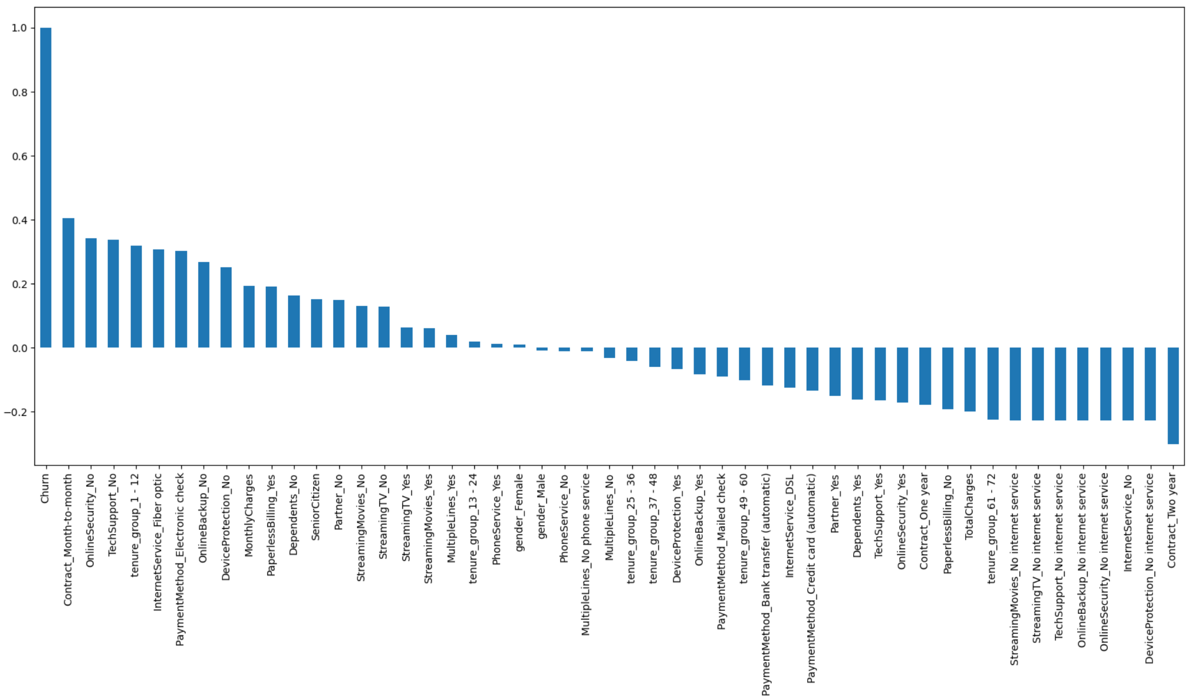
Surprising insight as higher Churn at lower Total Charges

More Analysis based on Multivariable

Building a corelation of all predictors with 'Churn'

In [ ]:
```
plt.figure(figsize=(20,8))
telecom_data_dummies.corr()['Churn'].sort_values(ascending = False).plot(kir
```

Out[ ]: <AxesSubplot:>



It tells us about which predictor gives more insight about Churn

Derived Insight:

HIGH Churn seen in case of Month to month contracts, No online security, No Tech support, First year of subscription and Fibre Optics Internet

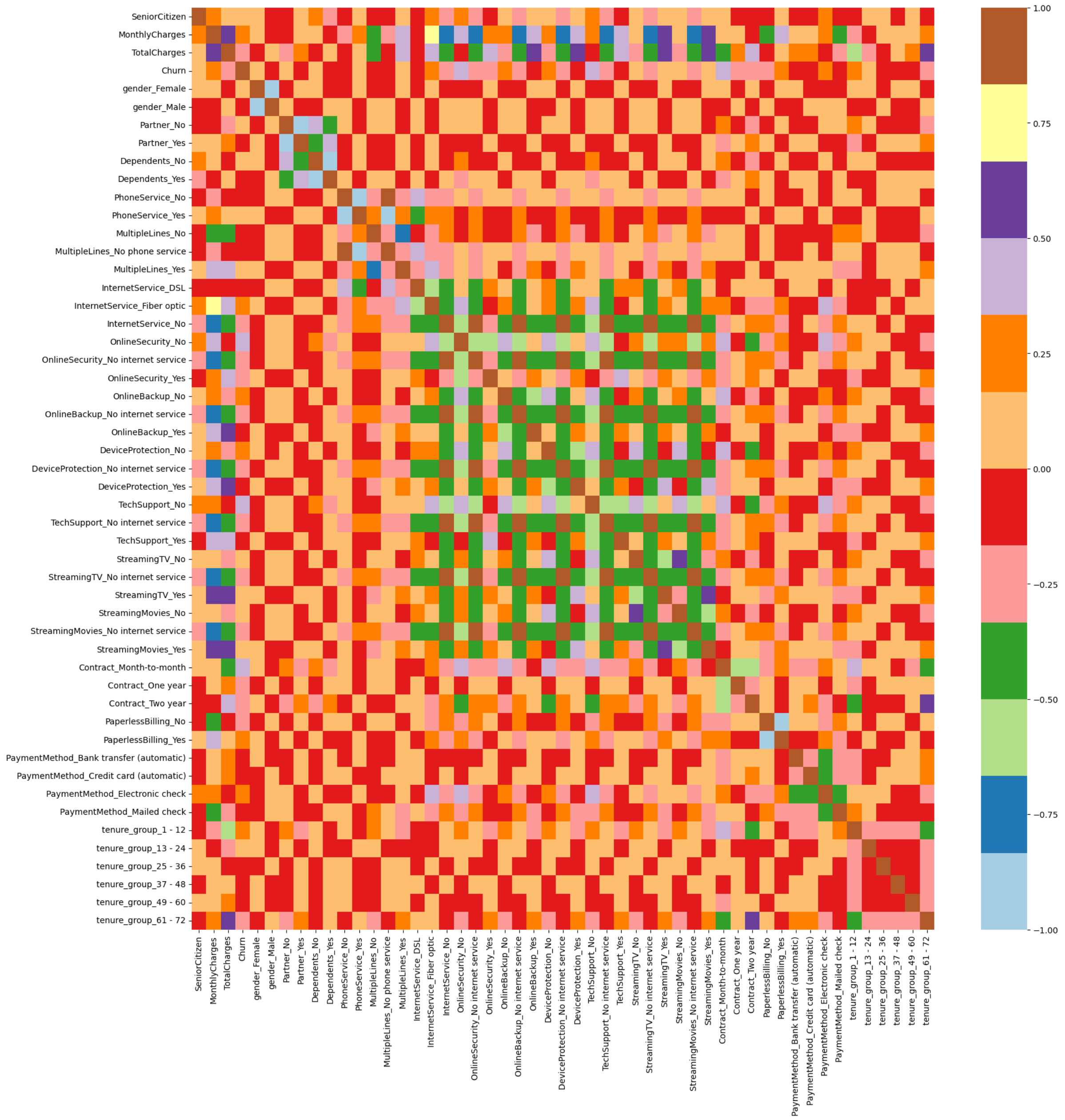
LOW Churn is seens in case of Long term contracts, Subscriptions without internet service and The customers engaged for 5+ years

Factors like Gender, Availability of PhoneService and # of multiple lines have alomost NO impact on Churn

Creating HeatMap

In [ ]:
```
plt.figure(figsize=(20,20))
sns.heatmap(telecom_data_dummies.corr(), cmap="Paired")
```

Out[ ]: <AxesSubplot:>



Bivariable Analysis

Creating a new Datafreame for Churners and one for Non Churners

In [ ]:
```
#creating new DataFrames
new_df1_target0 = telecom_data.loc[telecom_data["Churn"]==0]
new_df1_target1 = telecom_data.loc[telecom_data["Churn"]==1]
```

In [ ]:
```
#checking DataFrame
new_df1_target0.head()
```

Out[ ]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetService
0	Female	0	Yes	No	No	No phone service	DSL
1	Male	0	No	No	Yes	No	DSL
3	Male	0	No	No	No	No phone service	DSL
6	Male	0	No	Yes	Yes	Yes	Fiber optic
7	Female	0	No	No	No	No phone service	DSL

In [ ]:
```
new_df1_target1.head()
```

Out[ ]:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	InternetServic
2	Male	0	No	No	Yes	No	DS
4	Female	0	No	No	Yes	No	Fiber opt
5	Female	0	No	No	Yes	Yes	Fiber opt
8	Female	0	Yes	No	Yes	Yes	Fiber opt
13	Male	0	No	No	Yes	Yes	Fiber opt

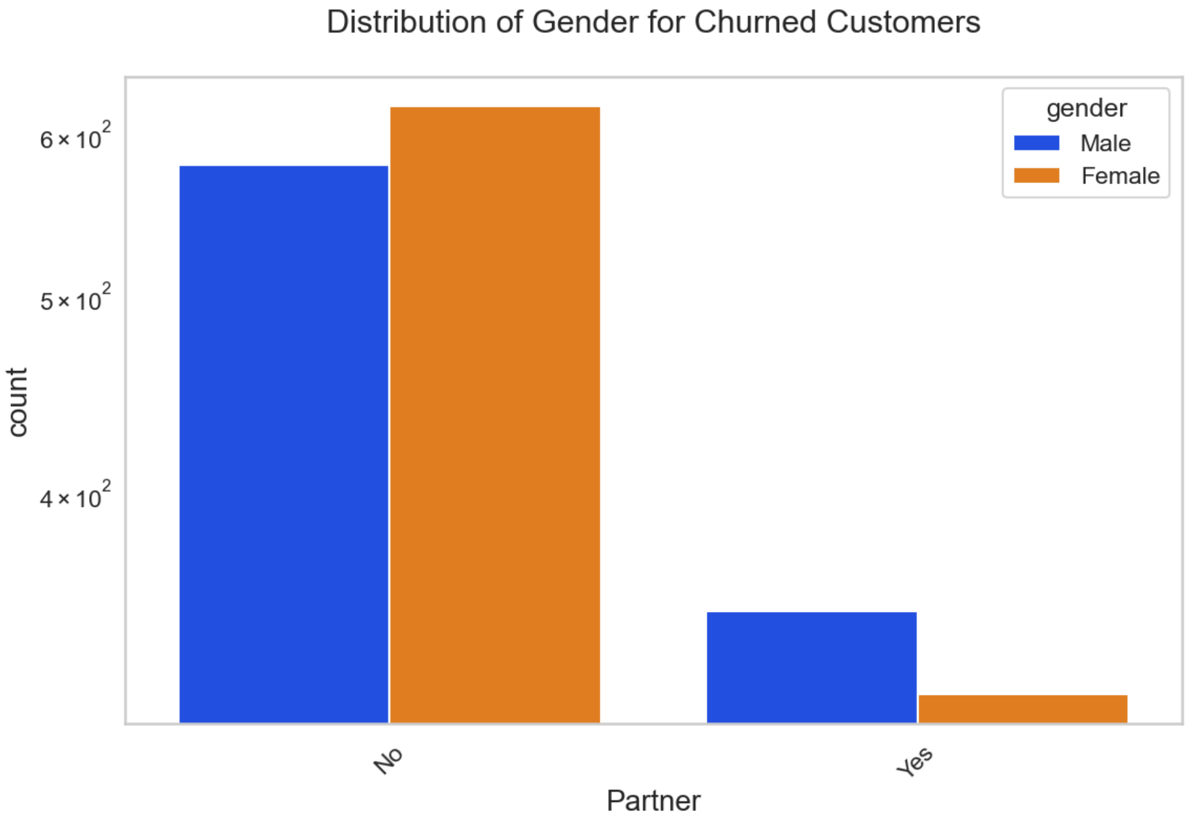
Creating a Function for ploting graph

In [ ]:
```
def uniplot(df,col,title,hue =None):

    sns.set_style('whitegrid')
    sns.set_context('talk')
    plt.rcParams["axes.labelsize"] = 20
    plt.rcParams['axes.titlesize'] = 22
    plt.rcParams['axes.titlepad'] = 30


    temp = pd.Series(data = hue)
    fig, ax = plt.subplots()
    width = len(df[col].unique()) + 7 + 4*len(temp.unique())
    fig.set_size_inches(width , 8)
    plt.xticks(rotation=45)
    plt.yscale('log')
```

```
    plt.title(title)
    ax = sns.countplot(data = df, x= col, order=df[col].value_counts().index

    plt.show()

print('Function Created Successfully')
```

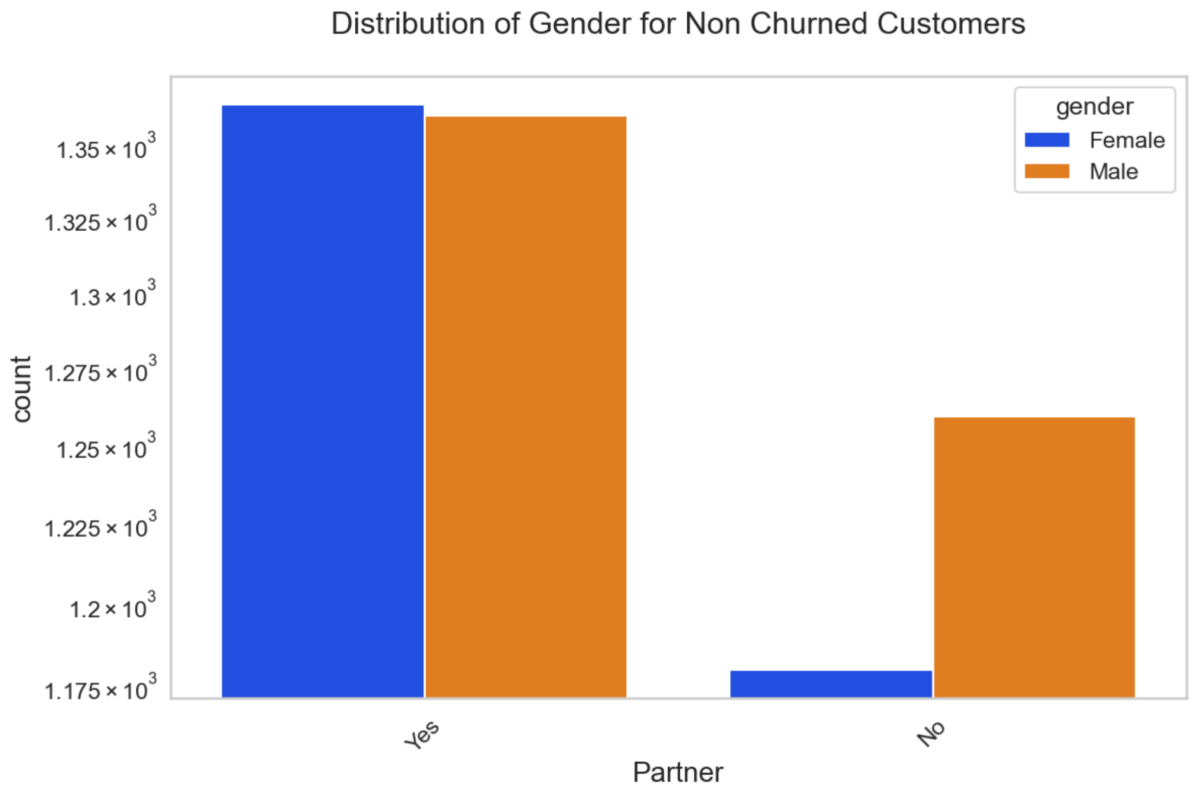
Function Created Successfully

In [ ]:
```
#calling Function
uniplot(new_df1_target1,col='Partner',title='Distribution of Gender for Chur
```

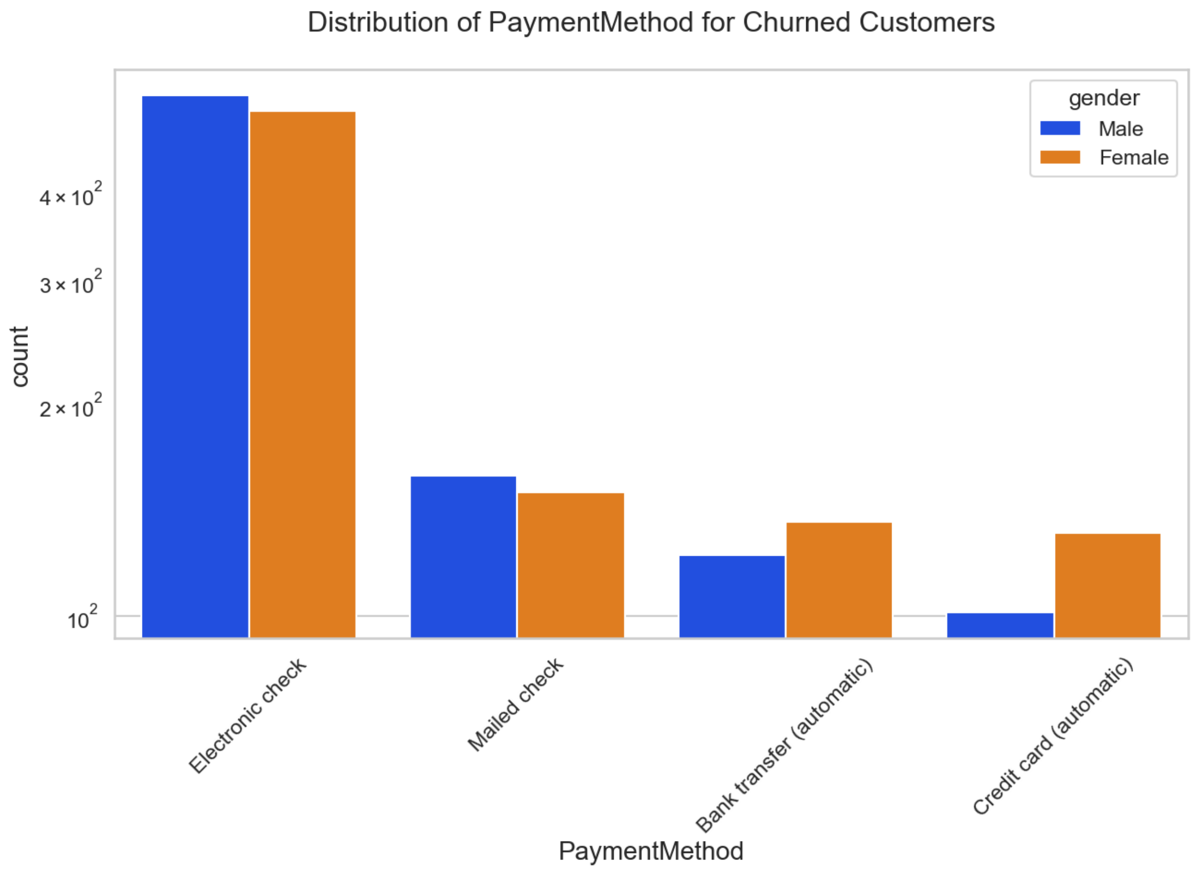


No much relevent insights can be drawn from above graph

In [ ]:
```
#for non Churners
uniplot(new_df1_target0,col='Partner',title='Distribution of Gender for Non
```

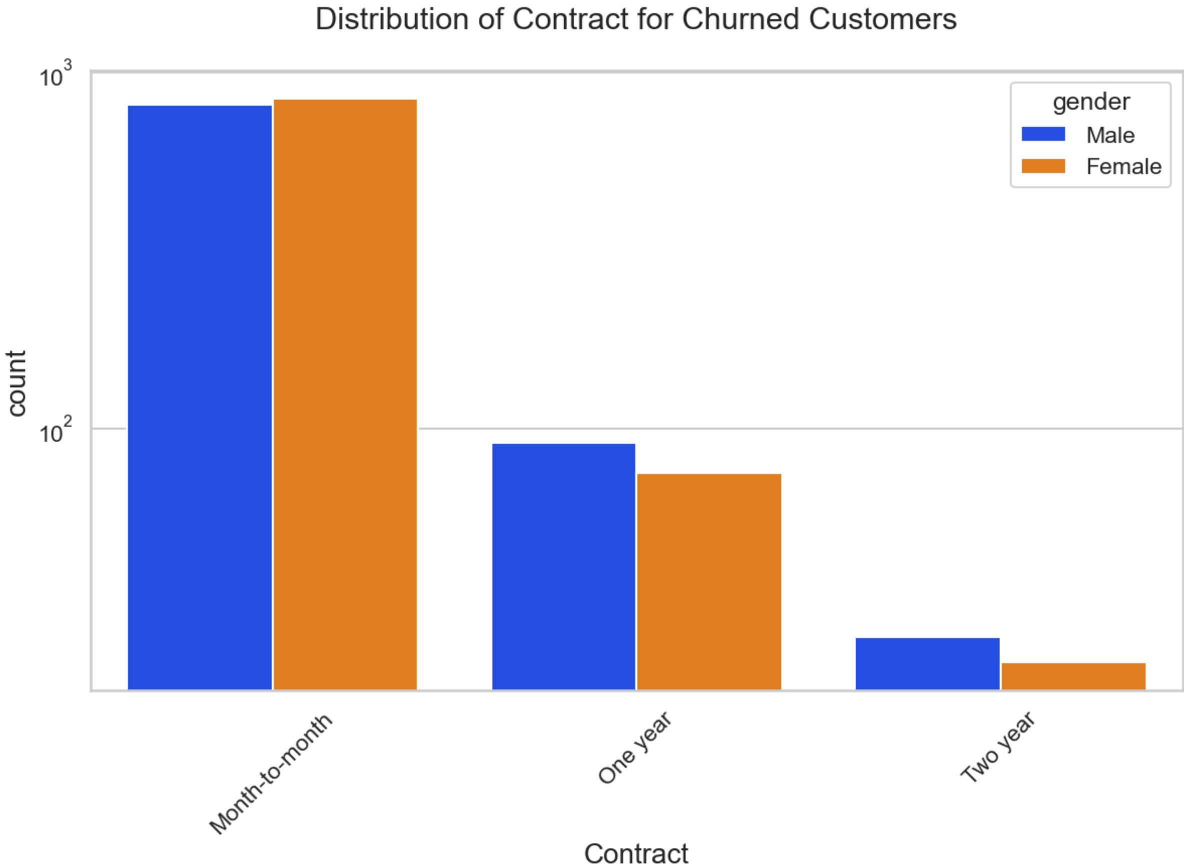



```
In [ ]: #for PaymentMethod
uniplot(new_df1_target1,col='PaymentMethod',title='Distribution of PaymentMe
```

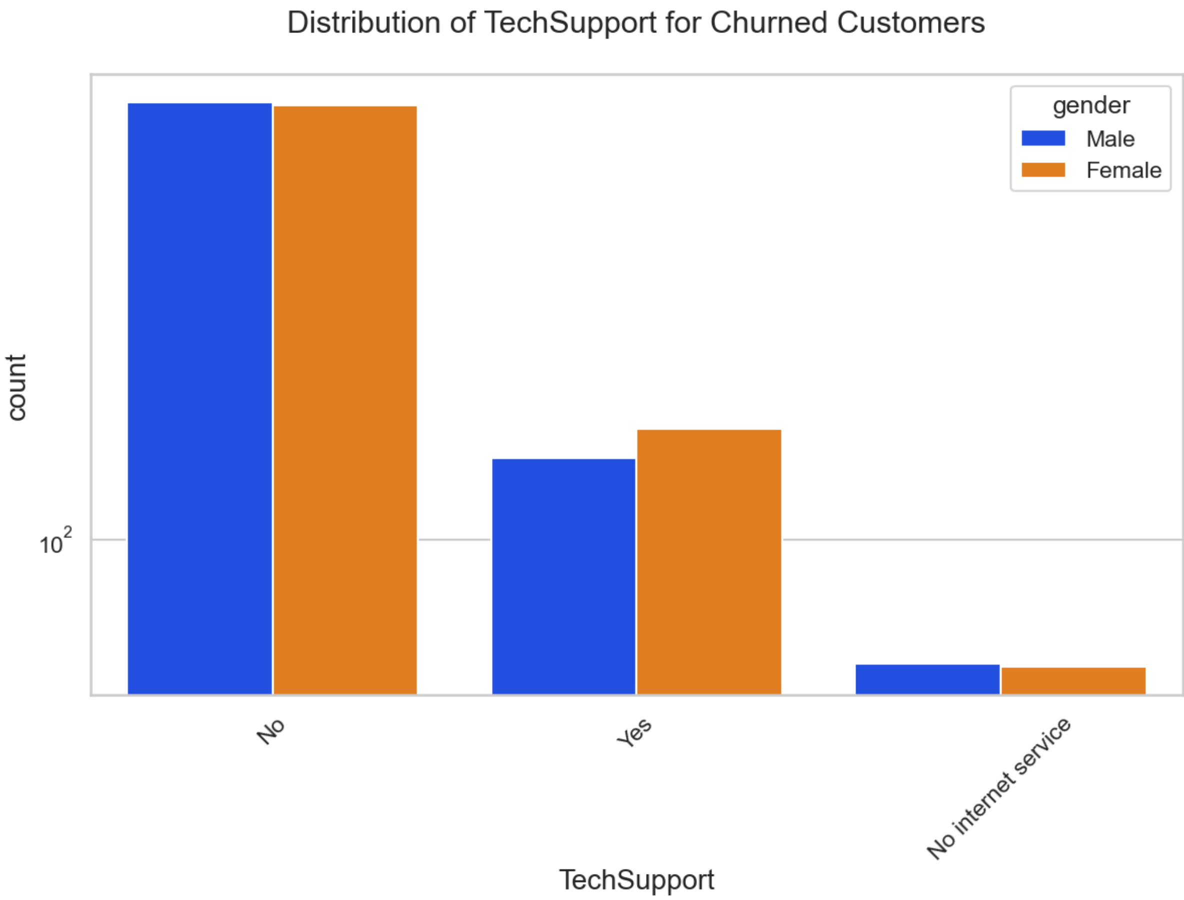


Insight from above graph is that : Most females with CreditCard are Churners

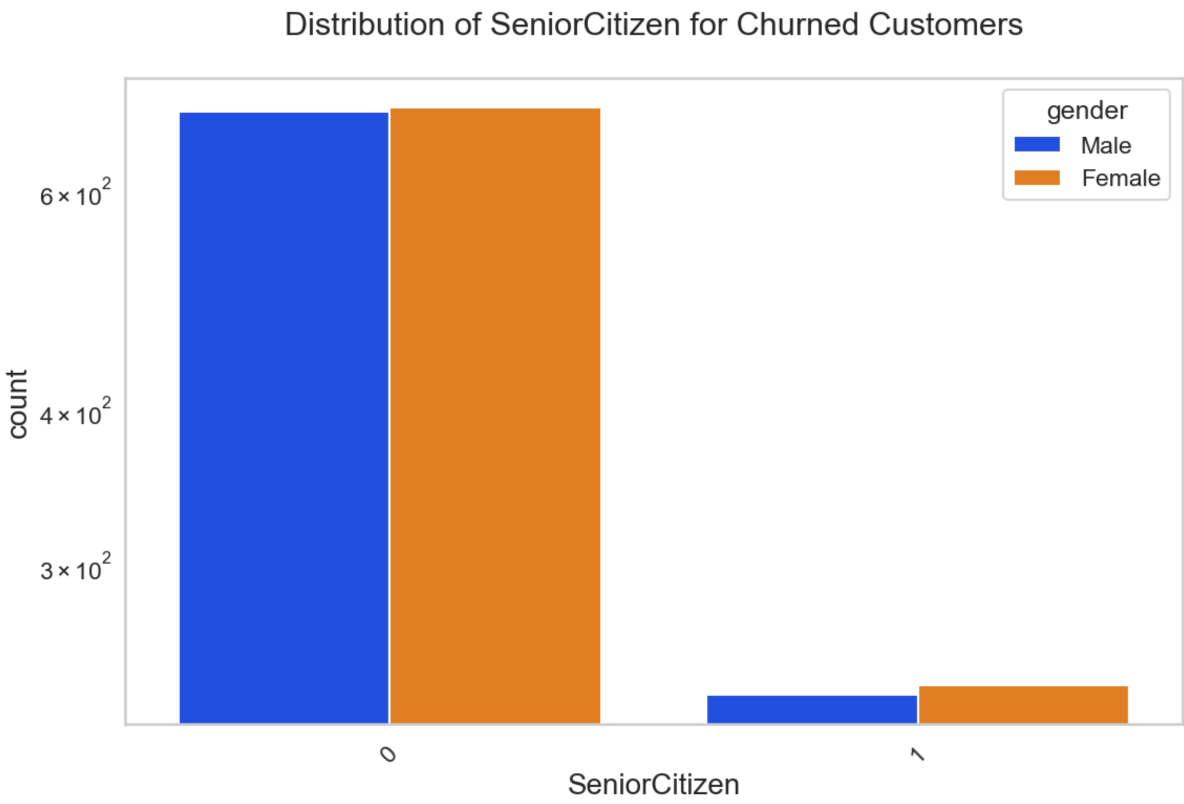
```
In [ ]: #for Contract
uniplot(new_df1_target1,col='Contract',title='Distribution of Contract for C
```



In [ ]:
```
#For Tech support
uniplot(new_df1_target1,col='TechSupport',title='Distribution of TechSupport
```



In [ ]:
```
#For SeniorCitizens
uniplot(new_df1_target1,col='SeniorCitizen',title='Distribution of SeniorCit
```



CONCLUSION These are some of the quick insights from this Dataset :

Electronic check medium are the highest churners Contract Type - Monthly customers are more likely to churn because of no contract terms, as they are free to go customers. No Online security, No Tech Support category are high churners Non senior Citizens are high churners

In [ ]:
```
#converting Dataset into csv file
telecom_data_dummies.to_csv('telecom_churn.csv')
print('File Created Successfully')
```

```
File Created Successfully
```

In [ ]: